Available online at http://scik.org

Commun. Math. Biol. Neurosci. 2021, 2021:71

https://doi.org/10.28919/cmbn/6159

ISSN: 2052-2541

FINGERPRINT IDENTIFICATION WITH MEMETIC ALGORITHM AND HIGH-PERFORMANCE COMPUTING MEMETIC ALGORITHM (HPCMA)

PRIATI ASSIROJ[1,2,*], HARCO LESLIE HENDRIC SPITS WARNARS[1], EDI ABDURACHMAN[1], ACHMAD IMAM KISTIJANTORO[3], ANTOINE DOUCET[4]

[1]Binus Graduate Program, Doctor of Computer Science, Bina Nusantara University, Jakarta 11530, Indonesia

[2]Manajemen Teknologi Keimigrasian, Politeknik Imigrasi, Depok 16514 Indonesia

[3]School of Electrical Engineering and Informatics, Institut Teknologi Bandung 40132, Indonesia

[4]Laboratoire L3i - Université de La Rochelle, Avenue Michel Crépeau, F-17 042 La Rochelle Cedex 1, France

Abstract: A high-performance computation (HPC) is a very needed thing in the organization, researcher, or government to increase innovation of the products and services. This term belongs to parallel computation that can be realized by utilizing multi-threads of the processor. Memetic algorithm (MA) is a simple and powerful algorithm that can run parallel. In this work, we use the original memetic algorithm and a parallel memetic algorithm, so-called a high-performance memetic algorithm (HPCMA) for fingerprint identification, one of the biometric features in the human body. To generate a high-quality biometric identification system, we should pay attention to accuracy, speed up, endurance, and acceptance uses reasonable resources, and reliable to prevent criminalities.

Keywords: memetic algorithm; high-performance memetic algorithm; biometric; fingerprint; speedup.

2010 AMS Subject Classification: 68W40.

*Corresponding author

E-mail address: priati@binus.ac.id

Received May 29, 2021

1. INTRODUCTION

1.1 HIGH-PERFORMANCE COMPUTATION

A high-performance computation (HPC) themes include mathematic, analysis, optimization, modeling and programming, compiler, vector architecture, and parallelization [1]. HPC succeeded to solve several problems in pattern recognition [2][3][4].

Several popular computation techniques are Computer Cluster, Supercomputer, Grid Computing, and Cloud Computing then we call these are High-Performance Computation System[5]. To implement a parallel computation, we need high-end hardware that provides several processors, and also the right operating system to split the computation load to all the processors. Initially, this is an expensive system that is only found at the huge computer system as a supercomputer. Fortunately, parallel computation is now can be found at the ordinary computation. A high-performance computation with parallel technique has been done massively to address the complexity of simulation, modeling, and data analytics quickly and effectively.

1.2 BIOMETRIC

Biometrics are special features of the human body. To identify an individual personally we can refer to the biometric, such as voice, face, retina, and fingerprint. Fingerprint identification of the most used biometric identification method [6][7]. The fingerprint is also the most studied biometric feature [8], and several algorithms have been proposed [9] for processing [10], classification [8], [11]. To reduce computation time when identifying, we should not compare the entire fingerprint with the classification method [12][13], and use the indexing algorithm [14][11], or use the parallel computation [15][16]. There are indexing algorithms to increase processing time proposed by [17][18] and [19]. The indexing method [17] is better than the other method [20][21][22].

For a reliable automatic fingerprint identification system, we need to pay attention to these several things [23]; precision refers to a lower error level rate to ensure accuracy and efficiency that refers to the lower identification time or time to find the proper fingerprint.

1.3 MEMETIC ALGORITHM

Memetic algorithm (MA) is a simple, flexible, and powerful algorithm [24][25] that generates

high-quality solutions in many challenging problems [26][27][28]. To reduce computation time in an optimization challenge that involves many variables, we need the right code and algorithm.

This memetic algorithm is very useful in the data mining, such as classification for text analysis [29][30]; in the bio-medics such as DNA simulation [31][32]; in the computer networking [33][34]; feature selection [35]; molecular simulation [36]; quantum chemistry [37]; forecasting [38]; spectroscopic analysis [39]; geophysics analysis [40]; drugs invention [41]; genomic study [42], and many more.

2. METHOD

This work conducted with 3 personal computers with Intel i7 6600U 2.6 GHz 4 core (up to 2.8 GB), 16GB RAM, SSD SanDisk M.2 2200 256GB for data learning logic background (HPCMA data learning); computer system with Intel i5 2540M 2.6 GHz 4 core, 16GB RAM, SSD Samsung 850 EVO 500GB for java swing desktop or data learning interface MA (MA data learning); and computer system with Intel i5 2430M 2.4 GHz 4 core. 8GB RAM, Samsung SSD 860 EVO 250GB as a database server for data training and data learning. These 3 computers communicate with each other through network architecture. Figure 1 below is the flowchart of the data learning identification process.

According to Figure 1, the process starts with a fingerprint image local search then converts to a string array and converts string array to binary code. The algorithm looks for appropriate binary code in the data training and compares with 100% similarity then saves the result in the temporary storage. Then the program divides binary code for similarity 90%, it means 5% for header, 90% for the body, and 5% for footer and program looks for appropriate binary code in the data training and compares with 90% similarity then save the result in the temporary storage. For similarity 80%, 10% for header, 80% for the body, and 10% for the footer, the program looks for appropriate binary code in the data training and compares with 90% similarity then saves the result in the temporary storage.

A similar process is conducted for similarity 70% and 60%, then we measure the total of data

training and save to the temporary storage, and show the result. Figure 2 below is the separation of fingerprint 100%, 90% body, 80% body, 70% body, and 60%.

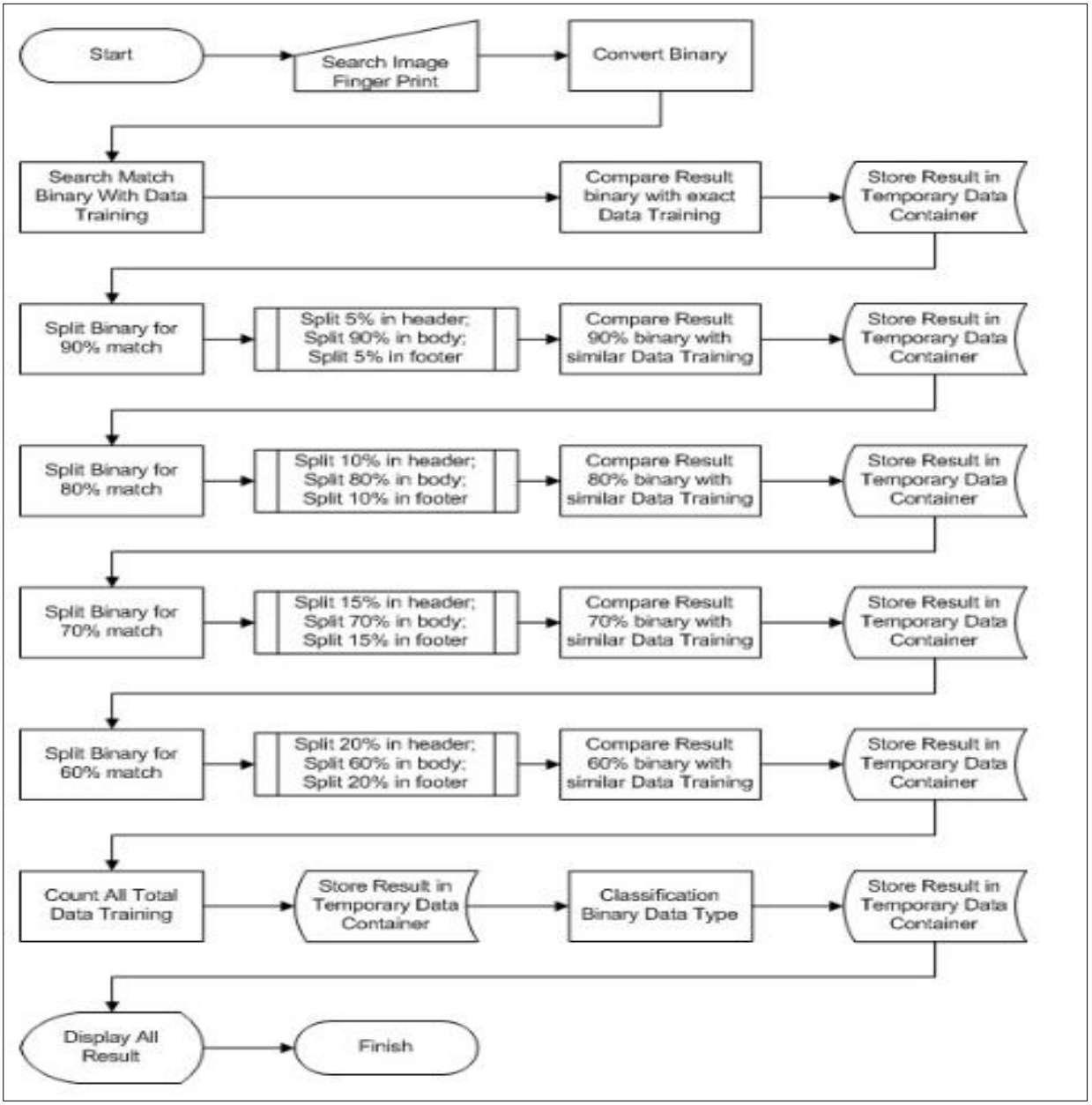

Figure 1. The flowchart of identification data learning

In this work, we use fingerprint data from FVC2006 that consists of 7200 fingerprints image data and categorized into 4 types or groups; partial type, full type, full type with white boundary, full and unclear type. Figure 3, 4, 5, and Figure 6 is the type of fingerprint.

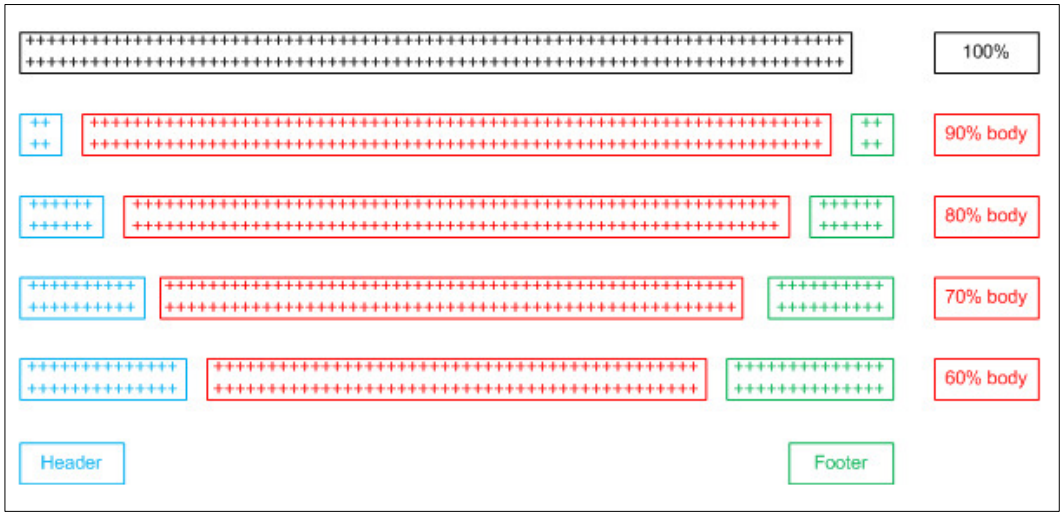

Figure 2. Header, body, and footer illustration

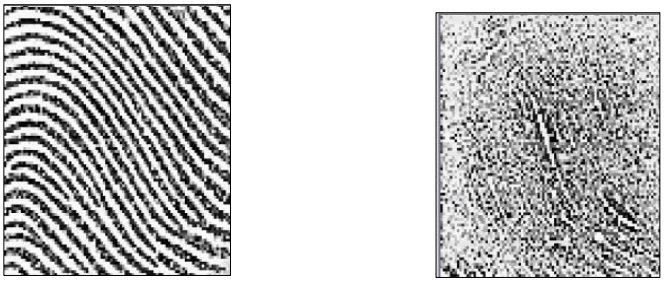

Figure 3. Partial fingerprint image

Figure 4. Full fingerprint image

Figure 5. Full fingerprint image with white boundary

Figure 6. Full and unclear fingerprint image

Then the fingerprint was divided into fifteen specimens. Each specimen consists of a group member of fingerprint types. Below, Figure 7, is the illustration of the specimen grouping and its members.


group 1 (1800 data)
group 2 (1800 data)
group 3 (1800 data)
group 4 (1800 data)
specimen 1
specimen 2
specimen 3
specimen 4
specimen 5
specimen 6
specimen 7
specimen 8
specimen 9
specimen 10
specimen 11
specimen 12
specimen 13
specimen 14

Figure 7. Data groups and specimen

The FVC2006 fingerprint dataset consists of 8 folders with a different characteristic fingerprint. From the 8 folders, we divide them into 4 folders that represent every fingerprint characteristic. Here are the folders:

a. DB1_A consists of 1680 fingerprint image data and DB1_B consists of 120 fingerprints image data. We have 1800 fingerprint images in this folder and this is Group 1.
b. DB2_A consists of 1680 fingerprint image data and DB2_B consists of 120 fingerprints image data. We have 1800 fingerprint images in this folder and this is Group 2.
c. DB3_A consists of 1680 fingerprint image data and DB3_B consists of 120 fingerprints image data. We have 1800 fingerprint images in this folder and this is Group 3.
d. DB4_A consists of 1680 fingerprint image data and DB4_B consists of 120 fingerprints image data. We have 1800 fingerprint images in this folder and this is Group 4.

According to Figure 7, Group 1, 2, 3, and Group 4 are illustrated in one circle that consists of 1800 fingerprint images, totally from these 4 folders we have 7200 fingerprints image files.

The blue circle is used group in each specimen. The green circle is new off-springs from the crossover process, 4900 new off-springs in total. The purple circle is new off-springs from the swap or mutation process, 4900 in total, thus totally, at the end of the algorithm process, we have 17000 fingerprints image files.

3. RESULT AND DISCUSSION

The experiment is conducted by collecting a sample from data learning randomly from each specimen. For example, we use 1_1.bmp from folder DB2_A. With a memetic algorithm, for 100% similarity, we find 1 identical file in 5380 milliseconds, and with HPCMA for 100% similarity, we find 1 identical file in 261 milliseconds. To obtain the speed up from each specimen, we compare serial processing time from the memetic algorithm with a parallel processing time from HPCMA. For 1_1.bmp the speed up is 206.3602 milliseconds.

For similarity 90%, with the memetic algorithm, we find 1 identical file in 291 milliseconds, and with HPCMA we find 1 identical file in 221 milliseconds, then the speedup is 2.357466 milliseconds. For similarity 80% we find 1 identical file in 463 milliseconds with a memetic algorithm, and with HPCMA we find 1 identical file in 197 milliseconds, then the speedup is 2.350254 milliseconds.

For similarity 70% with the same sample file, we find 1 identical file in 405 milliseconds with a memetic algorithm, and with HPCMA we find 1 identical file in 176 milliseconds, then the speedup is 2.301136 milliseconds. For similarity 60% with the memetic algorithm, we find 1 identical file in 352 milliseconds, and with HPCMA we find 1 identical file in 148 milliseconds, then the speedup is 2.378378 milliseconds. Table 1 below shows the result of the experiment.

Table 1. Experiment result

Spc.	Folder	File	Memetic Algorithm (MA)							HPCMA							Speed up HPCMA				
			Time for each similarity (ms)					File found	Total time	Time for each similarity (ms)					File found	Total time	Similarity				
			100%	90%	80%	70%	60%		(ms)	100%	90%	80%	70%	60%		(ms)	100%	90%	80%	70%	60%
1	DB2_A	1_1.bmp	53860	521	463	405	352	1	55601	261	221	197	176	148	1	261	206.3602	2.357466	2.350254	2.301136	2.378378
2	DB1_A	1_2.bmp	244	48	51	44	76	1	463	26	32	30	2	36	1	36	9.384615	1.5	1.7	22	2.111111
3	DB2_A	1_2.bmp	58836	526	446	405	343	1	60556	266	234	213	186	160	1	266	221.188	2.247863	2.093897	2.177419	2.14375
4	DB3_A	1_2.bmp	60093	469	409	360	296	1	61625	210	195	173	152	136	1	210	286.1571	2.405128	2.364162	2.368421	2.176471
5	DB4_A	1_2.bmp	26761	271	252	214	194	1	27692	122	112	102	92	83	1	122	219.3525	2.419643	2.470588	2.326087	2.337349
6	DB2_A	1_3.bmp	58882	506	477	398	352	1	60615	255	238	214	179	163	1	255	230.9098	2.12605	2.228972	2.223464	2.159509
7	DB3_A	1_3.bmp	53516	463	435	379	310	1	55103	221	199	183	165	164	1	221	242.1538	2.326633	2.377049	2.29697	1.890244
8	DB4_A	1_3.bmp	25062	256	235	212	180	1	25945	123	113	102	92	85	1	123	203.7561	2.265487	2.303922	2.304348	2.117647
9	DB3_A	1_3.bmp	53516	463	435	379	310	1	55103	221	199	183	165	164	1	221	242.1538	2.326633	2.377049	2.29697	1.890244
10	DB2_A	1_3.bmp	58882	506	477	398	352	1	60615	255	238	214	179	163	1	255	230.9098	2.12605	2.228972	2.223464	2.159509
11	DB3_A	1_3.bmp	53516	463	435	379	310	1	25945	221	199	183	165	164	1	221	242.1538	2.326633	2.377049	2.29697	1.890244
12	DB2_A	1_4.bmp	58496	526	479	420	353	1	60274	238	223	196	173	148	1	238	245.7815	2.358744	2.443878	2.427746	2.385135
13	DB2_A	1_4.bmp	58496	526	479	420	353	1	60274	238	223	196	173	148	1	238	245.7815	2.358744	2.443878	2.427746	2.385135
14	DB3_A	1_4.bmp	56395	481	413	356	331	1	57976	221	203	183	167	150	1	221	255.181	2.369458	2.256831	2.131737	2.206667
15	DB4_A	1_4.bmp	22756	267	238	206	191	1	23658	137	122	113	93	83	1	137	166.1022	2.188525	2.106195	2.215054	2.301205

Table 2. Speed up and efficiency of each similarity

Speed Up (milliseconds)					Number of *Threads*	**Efficiency**				
Similarity						*Similarity*				
100%	**90%**	**80%**	**70%**	**60%**		**100%**	**90%**	**80%**	**70%**	**60%**
206.3602	2.357466	2.350254	2.301136	2.378378	5	41.27203065	0.471493213	0.470050761	0.460227273	0.475675676
9.384615	1.5	1.7	22	2.111111	5	1.876923077	0.3	0.34	4.4	0.422222222
221.188	2.247863	2.093897	2.177419	2.14375	5	44.23759398	0.44957265	0.418779343	0.435483871	0.42875
286.1571	2.405128	2.364162	2.368421	2.176471	5	57.23142857	0.481025641	0.47283237	0.473684211	0.435294118
219.3525	2.419643	2.470588	2.326087	2.337349	5	43.8704918	0.483928571	0.494117647	0.465217391	0.46746988
230.9098	2.12605	2.228972	2.223464	2.159509	5	46.18196078	0.425210084	0.445794393	0.444692737	0.43190184
242.1538	2.326633	2.377049	2.29697	1.890244	5	48.43076923	0.465326633	0.475409836	0.459393939	0.37804878
203.7561	2.265487	2.303922	2.304348	2.117647	5	40.75121951	0.453097345	0.460784314	0.460869565	0.423529412
242.1538	2.326633	2.377049	2.29697	1.890244	5	48.43076923	0.465326633	0.475409836	0.459393939	0.37804878
230.9098	2.12605	2.228972	2.223464	2.159509	5	46.18196078	0.425210084	0.445794393	0.444692737	0.43190184
242.1538	2.326633	2.377049	2.29697	1.890244	5	48.43076923	0.465326633	0.475409836	0.459393939	0.37804878
245.7815	2.358744	2.443878	2.427746	2.385135	5	49.15630252	0.471748879	0.48877551	0.485549133	0.477027027
245.7815	2.358744	2.443878	2.427746	2.385135	5	49.15630252	0.471748879	0.48877551	0.485549133	0.477027027
255.181	2.369458	2.256831	2.131737	2.206667	5	51.0361991	0.473891626	0.45136612	0.426347305	0.441333333
166.1022	2.188525	2.106195	2.215054	2.301205	5	33.22043796	0.437704918	0.421238938	0.443010753	0.460240964

From table 1 above, we know the processing time from the memetic algorithm is slower than HPCMA. We also measure the speed up, which becomes one of the goals of parallel computation [43]. Table 2 below is a detail of the speedup and efficiency of HPCMA in each specimen and each similarity, and Figure 8 is a visualization of efficiency for each similarity.

According to Table 2, for every similarity index, we use 5 threads in the data learning experiment. The efficiency of HPCMA is a comparison of speed up and threads number. From table 2, the efficiency from similarity 100% is bigger than the other similarities.

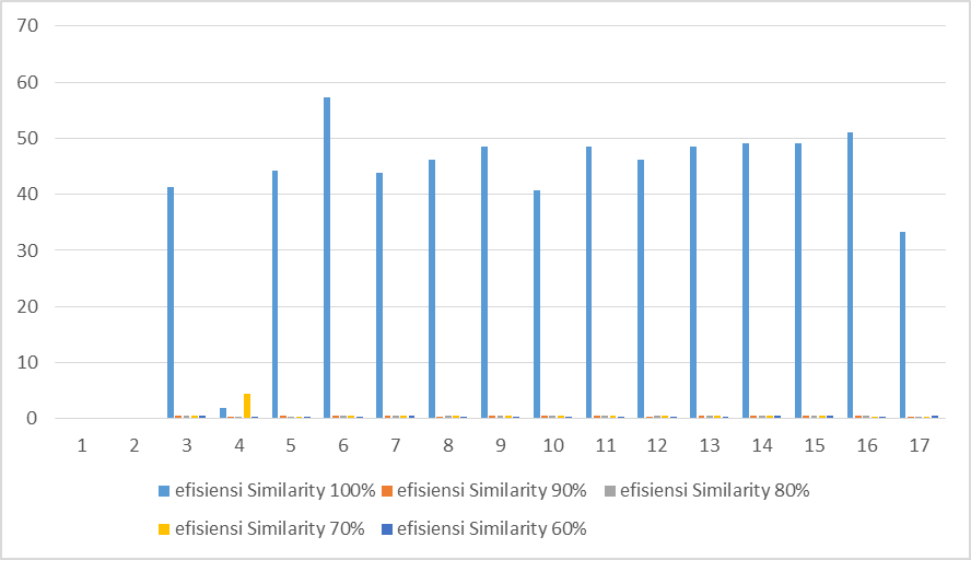

Figure 8. Efficiency for each similarity

The better identification process is when we obtain the smaller efficiency because it was more efficient to find the fingerprint in the database [23], thus the HPCMA is quite efficient to process these similarities.

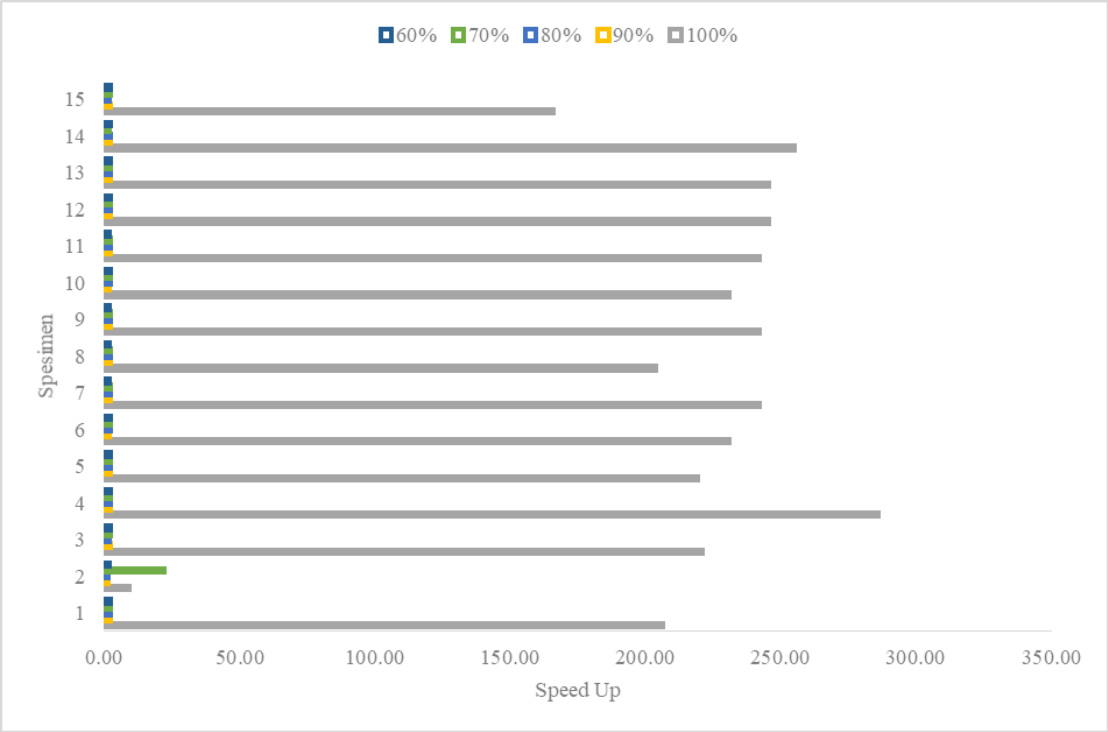

Figure 9. speed up for every specimen in each similarity

Figure 9 shows the speedup for every similarity, 100%, 90%, 80%, 70%, and 60%. For specimen 1, speed up for similarity 100% is 206.3602 milliseconds, for similarity 90% is 2.357466 milliseconds, for similarity 80% is 2.350254 milliseconds, for similarity 70% is 2.301136 milliseconds, and the speed up for similarity 60% is 2.378378 milliseconds. The same thing occurred to specimen 2 to specimen 15, the higher similarity is higher speed up.

4. CONCLUSION

The identification process is related to the ability to find the similarities of the object. To obtain a high similarity index, in this work, we make a partition to the fingerprint image, header, body, and footer as illustrated in Figure 2, then design the similarity index in 100%, 90%, 80%, 70%, and 60%, and measure the processing time of each similarity.

From several experimental samples with memetic algorithm and HPCMA, generally in 100%, the processing time is longer than the other. The processing time of the memetic algorithm is 3345766

milliseconds and HPCMA is 72904 milliseconds. Based on the performance HPCMA is faster than the original MA and more efficient.

In future work, we can train the model of memetic algorithm and HPCMA in many image data and in many platforms using GPU or raspberry.

ACKNOWLEDGEMENT

This work is supported by Research and Technology Transfer Office, Bina Nusantara University as a part of Bina Nusantara University's International Research Grant entitled MEMETIC ALGORITHM IN HIGH-PERFORMANCE COMPUTATION with contract number: No.026/VR.RTT/IV/2020 and contract date: 6 April 2020.

CONFLICT OF INTERESTS

The author(s) declare that there is no conflict of interests.

REFERENCES

[1] P. Marksteiner, High-performance computing — an overview, Computer Phys. Commun. 97 (1996), 16–35.

[2] M.S. Seidenberg, J.L. McClelland, A distributed, developmental model of word recognition and naming, Psychol. Revi. 96 (1989), 523–568.

[3] A. Datta, S. Soundaralakshmi, Fast parallel algorithm for distance transforms, in: Proceedings 15th International Parallel and Distributed Processing Symposium. IPDPS 2001, IEEE Comput. Soc, San Francisco, CA, USA, 2001: pp. 1130–1134.

[4] A. Stamatakis, M. Ott, Exploiting Fine-Grained Parallelism in the Phylogenetic Likelihood Function with MPI, Pthreads, and OpenMP: A Performance Study, in: M. Chetty, A. Ngom, S. Ahmad (Eds.), Pattern Recognition in Bioinformatics, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008: pp. 424–435.

[5] P. Assiroj, H.L.H.S. Warnars, R. Kosala, B. Ranti, S. Supangat, A.I. Kistijantoro, E. Abdurrachman, The Form of High-Performance Computing: A Survey, IOP Conf. Ser.: Mater. Sci. Eng. 662 (2019), 052002.

[6] A. K. Jain, P. Flynn, and A. A. Ross, Handbook of Biometrics, Springer, New York, 2007.

[7] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, Handbook of Fingerprint Recognition, 2nd ed. Springer London, 2009.

[8] M. Galar, J. Derrac, D. Peralta, I. Triguero, et al. A survey of fingerprint classification Part I: Taxonomies on feature extraction methods and learning models, Knowl.-Based Syst. 81 (2015), 76–97.

[9] K. Cao, E. Liu, A.K. Jain, Segmentation and Enhancement of Latent Fingerprints: A Coarse to Fine RidgeStructure Dictionary, IEEE Trans. Pattern Anal. Mach. Intell. 36 (2014), 1847–1859.

[10] A. Kumar, C. Kwong, Towards Contactless, Low-Cost and Accurate 3D Fingerprint Identification, in: 2013 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Portland, OR, USA, 2013: pp. 3438–3443.

[11] R. Cappelli, M. Ferrara, D. Maltoni, Fingerprint Indexing Based on Minutia Cylinder-Code, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2011), 1051–1057.

[12] R. Cappelli, D. Maio, The State of the Art in Fingerprint Classification, in: N. Ratha, R. Bolle (Eds.), Automatic Fingerprint Recognition Systems, Springer-Verlag, New York, 2004: pp. 183–205.

[13] J.-H. Hong, J.-K. Min, U.-K. Cho, S.-B. Cho, Fingerprint classification using one-vs-all support vector machines dynamically ordered with naıve Bayes classifiers, Pattern Recognit. 41 (2008), 662–671.

[14] B. Bhanu, X. Tan, A Triplet Based Approach for Indexing of Fingerprint Database for Identification, in: J. Bigun, F. Smeraldi (Eds.), Audio- and Video-Based Biometric Person Authentication, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001: pp. 205–210.

[15] H.H. Le, N.H. Nguyen, T.T. Nguyen, Exploiting GPU for Large Scale Fingerprint Identification, in: N.T. Nguyen, B. Trawiński, H. Fujita, T.-P. Hong (Eds.), Intelligent Information and Database Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016: pp. 688–697.

[16] D. Peralta, I. Triguero, R. Sanchez-Reillo, F. Herrera, J.M. Benitez, Fast fingerprint identification for large databases, Pattern Recognit. 47 (2014), 588–602.

[17] E. Chávez, G. Navarro, A compact space decomposition for effective metric indexing, Pattern Recognit. Lett. 26 (2005), 1363–1376.

[18] E. Chávez, G. Navarro, R. Baeza-Yates, J.L. Marroquín, Searching in metric spaces, ACM Comput. Surv. 33 (2001), 273–321.

[19] G. Navarro, R. Uribe-Paredes, Fully dynamic metric access methods based on hyperplane partitioning, Inform.

Syst. 36 (2011), 734–747.

[20] R.J. Barrientos, J.I. Gómez, C. Tenllado, M.P. Matias, M. Marin, kNN Query Processing in Metric Spaces Using GPUs, in: E. Jeannot, R. Namyst, J. Roman (Eds.), Euro-Par 2011 Parallel Processing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011: pp. 380–392.

[21] V. Gil-Costa, R. Barrientos, M. Marin, C. Bonacic, Scheduling Metric-Space Queries Processing on Multi-Core Processors, in: 2010 18th Euromicro Conference on Parallel, Distributed and Network-Based Processing, IEEE, Pisa, 2010: pp. 187–194.

[22] M. Marin, V. Gil-Costa, C. Bonacic, R. Baeza-Yates, I.D. Scherson, Sync/Async parallel search for the efficient design and construction of web search engines, Parallel Comput. 36 (2010), 153–168.

[23] G.K. Manacher, Production and Stabilization of Real-Time Task Schedules, J. ACM. 14 (1967), 439–465.

[24] P. Merz, B. Freisleben, Fitness landscapes and memetic algorithm design. In: D. Corne, M. Dorigo and F. Glover (eds.), New Ideas in Optimization. McGraw-Hill, 1999: pp. 245–260.

[25] Y.S. Ong, M.H. Lim, N. Zhu, K.W. Wong, Classification of adaptive memetic algorithms: a comparative study, IEEE Trans. Syst., Man, Cybern. B. 36 (2006), 141–152.

[26] A. Caponio, G.L. Cascella, F. Neri, N. Salvatore, M. Sumner, A Fast Adaptive Memetic Algorithm for Online and Offline Control Design of PMSM Drives, IEEE Trans. Syst., Man, Cybern. B. 37 (2007), 28–41.

[27] L. Jiao, M. Gong, S. Wang, B. Hou, Z. Zheng, Q. Wu, Natural and Remote Sensing Image Segmentation Using Memetic Computing, IEEE Comput. Intell. Mag. 5 (2010), 78–91.

[28] M. Urselmann, S. Barkmann, G. Sand, S. Engell, A Memetic Algorithm for Global Optimization in Chemical Process Synthesis Problems, IEEE Trans. Evol. Computat. 15 (2011), 659–683.

[29] Y. Lu, S. Wang, S. Li, C. Zhou, Particle swarm optimizer for variable weighting in clustering high-dimensional data, Mach. Learn. 82 (2011), 43–70.

[30] L. Bai, J. Liang, C. Dang, F. Cao, A novel attribute weighting algorithm for clustering high-dimensional categorical data, Pattern Recognit. 44 (2011), 2843–2861.

[31] S. Lang, P. Drouvelis, E. Tafaj, P. Bastian, B. Sakmann, Fast extraction of neuron morphologies from large-scale SBFSEM image stacks, J. Comput Neurosci. 31 (2011), 533–545.

[32] S. Bahmann, J. Kortus, EVO—Evolutionary algorithm for crystal structure prediction, Computer Phys. Commun.

184 (2013), 1618–1625.

[33] K.R. Varshney, A.S. Willsky, Linear Dimensionality Reduction for Margin-Based Classification: High-Dimensional Data and Sensor Networks, IEEE Trans. Signal Process. 59 (2011), 2496–2512.

[34] H. Ergun, D. Van Hertem, R. Belmans, Transmission System Topology Optimization for Large-Scale Offshore Wind Integration, IEEE Trans. Sustain. Energy. 3 (2012), 908–917.

[35] J.-H. Hong, S.-B. Cho, Efficient huge-scale feature selection with speciated genetic algorithm, Pattern Recognit. Lett. 27 (2006), 143–150.

[36] Y. Zhao, F.K. Sheong, J. Sun, P. Sander, X. Huang, A fast parallel clustering algorithm for molecular simulation trajectories, J. Comput. Chem. 34 (2013), 95–104.

[37] Y. Shao, L.F. Molnar, Y. Jung, et al. Advances in methods and algorithms in a modern quantum chemistry program package, Phys. Chem. Chem. Phys. 8 (2006), 3172–3191.

[38] D. Niu, Y. Wang, D.D. Wu, Power load forecasting using support vector machine and ant colony optimization, Expert Syst. Appl. 37 (2010), 2531–2539.

[39] T.K. Roy, R.B. Gerber, Vibrational self-consistent field calculations for spectroscopy of biological molecules: new algorithmic developments and applications, Phys. Chem. Chem. Phys. 15 (2013), 9468.

[40] J. Blum, F.X. Le Dimet, I.M. Navon, Data Assimilation for Geophysical Fluids. Handbook of Numerical Analysis, vol. XIV. Elsevier, Amsterdam, (2005).

[41] J.T. Dudley, E. Schadt, M. Sirota, A.J. Butte, E. Ashley, Drug Discovery in a Multidimensional World: Systems, Patterns, and Networks, J. of Cardiovasc. Trans. Res. 3 (2010), 438–447.

[42] W. Shi, G. Wahba, R.A. Irizarry, H.C. Bravo, S.J. Wright, The partitioned LASSO-patternsearch algorithm with application to gene expression data, BMC Bioinform. 13 (2012), 98.

[43] V. Pachori, G. Ansari, and N. Chaudhary, Improved performance of advance encryption standard using parallel computing, Int. J. Eng. Res. Appl. 2 (2012), 967–971.